2012

Planar Polygon Extraction and Merging from Depth Images

Joydeep Biswas
University of Massachusetts Amherst

Manuela M. Veloso
Carnegie Mellon University

Planar Polygon Extraction and Merging from Depth Images

Joydeep Biswas
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
joydeepb@ri.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
mmv@cs.cmu.edu

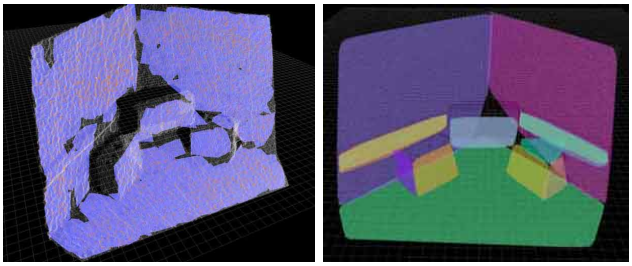
***Abstract*— There has been considerable interest recently in building 3D maps of environments using inexpensive depth cameras like the Microsoft Kinect sensor. We exploit the fact that typical indoor scenes have an abundance of planar features by modeling environments as sets of plane polygons. To this end, we build upon the Fast Sampling Plane Filtering (FSPF) algorithm that extracts points belonging to local neighborhoods of planes from depth images, even in the presence of clutter. We introduce an algorithm that uses the FSPF-generated plane filtered point clouds to generate convex polygons from individual observed depth images. We then contribute an approach of merging these detected polygons across successive frames while accounting for a complete history of observed plane filtered points without explicitly maintaining a list of all observed points. The FSPF and polygon merging algorithms run in real time at full camera frame rates with low CPU requirements: in a real world indoor environment scene, the FSPF and polygon merging algorithms take 2.5 ms on average to process a single 640 $\times$ 480 depth image. We provide experimental results demonstrating the computational efficiency of the algorithm and the accuracy of the detected plane polygons by comparing with ground truth.**

I. Introduction

With the availability of low-cost depth cameras like the Microsoft Kinect sensor, there has been renewed interest in building 3D models of indoor environments. Much of the current work towards the goal of 3D mapping (*e.g.* [1], [2]) have focused on generating detailed, dense 3D models of environments. However, typical indoor environments consist of large planar features that could instead be used to build simplified models of indoor environments at a fraction of the memory requirements of dense 3D models. Hence, we are interested in modeling such indoor environments by extracting the dominant planar features. This could be performed as a post-processing step by mesh simplification [3] of a detailed 3D model, or (as we propose) in an incremental, on-line manner, where for every observed depth image, only planar regions are extracted to be merged into the model of the environment. Our approach can therefore be broken down into the following steps:

1) For every observed depth image, compute polygons to best approximate the dominant planes in that image.
2) Find correspondences between the polygons in the current image with the polygons in the map.
3) Iteratively merge corresponding polygons from successive images, thus growing the map.

This work was supported by the National Science Foundation award number NSF IIS-1012733. The views and conclusions contained in this document are those of the authors only.



(a) (b)

Fig. 1: Results from a test scene: (a) Plane Filtered points shown as orange points, corresponding convex polygons shown in blue. The complete 3D point cloud is overlaid as translucent grey for reference. (b) Map generated by merging polygons from multiple depth images

Our approach uses the Fast Sampling Plane Filtering (FSPF) algorithm [4], which extracts points belonging to local neighborhoods of planes from depth images, even in the presence of clutter. Given the FSPF generated neighborhoods of local planar points, we first compute the plane parameters of each neighborhood by eigenvector decomposition of the scatter matrix, and then construct a convex hull over the points to define the polygon boundary. The scatter matrix for each polygon is stored in a decoupled manner which, as we shall show, allows us to merge polygons over successive observations and compute the least square fit over *all* observed points over time without having to explicitly maintain a list of them. We introduce an image render based ray-casting method for determining correspondences between the polygons observed in the latest depth image, to the polygons in the map. Polygons in the latest depth image that do not correspond to any existing polygons in the map are added as new polygons to the map.

To illustrate the key processed results of polygon construction and merging, Fig. 1 shows the plane filtered points and polygons from a single frame, and the map obtained by merging 15 depth image frames from a sample scene.

We provide experimental results demonstrating the accuracy of the polygon estimation and merging algorithms when compared to ground truth, and also provide results from a number of differing indoor scenes.

II. Related Work

The problem of plane (and general geometric shape) detection from raw 3D point clouds has been studied in the past(*e.g.* [5], [6]), where the plane (or general shape) detection is performed offline, typically as a post-processing step on the complete 3D point cloud. On the other hand, real-time online approaches that use data from depth images can exploit the fact that depth cameras make observations in "2.5D": the depth values are observed on a (virtual) 2D image plane originating from a single point. Region growing [7] exploits the local correlation in the depth image and attempts to assign planes to every 3D point. The Fast Sampling Plane Filtering algorithm (FSPF) [4] instead samples points at random and uses RANSAC [8] in local neighborhoods.

A number of 3D mapping algorithms (*e.g.*, [9], [10], [11]) have been developed that build maps using 3D points in space. An alternative approach to mapping using the raw 3D points is to map using planar features extracted from the 3D point cloud (*e.g.*, [12], [13]). Approaches to extraction of geometric features in point clouds include 3D Hough transform [14] and region growing [7]. In particular, 3D Plane SLAM [15] is a 6D SLAM algorithm that uses observed 3D point clouds to construct maps with 3D planes. In this approach, plane detection relies on region growing for plane extraction, whereas our approach uses local RANSAC filtering of the depth image.

Recent approaches have enabled dense 3D reconstruction of environments using a single moving RGB camera [2] or using depth only [16]. The resultant maps of these approaches are intended to be detailed models of the environment, whereas our approach aims at building simplified models of the environment.

III. Plane Polygon Construction And Merging

Given an observed raw depth image I, FSPF creates a list P of n 3D points that belong to local planes and a list R of the associated plane normals.

A. Polygon Construction

From the plane filtered point cloud P, local neighborhoods of points are merged to form local convex polygons. A convex polygon is denoted by the tuple $c = \{\hat{P}, n, \bar{p}, r, b_1, b_2, B\}$ where $\hat{P}$ is the set of 3D points used to construct the convex polygon, n the number of points in $\hat{P}$, $\bar{p}$ the centroid of the polygon, r the normal to the polygon plane, b_1 and b_2 the 2D basis vectors on the plane of the polygon and B the set of 3D points that define the convex boundary of the polygon. Given a neighborhood of plane filtered 3D points $\hat{P}$, a convex polygon c is constructed from $\hat{P}$ as follows:

1) The polygon centroid $\bar{p}$ is computed as $\bar{p} = \frac{1}{n}\sum_{p_i \in \hat{P}} p_i$.
2) The scatter matrix S of the points in $\hat{P}$ is computed as $S = \sum_{p_i \in \hat{P}} (p_i - \bar{p})(p_i - \bar{p})^T$.
3) The normal r is then given by the eigenvector of S corresponding to its smallest eigenvalue.
4) The orthogonal basis vectors that span the plane, b_1 and b_2 are the remaining two eigenvectors of S.
5) The boundary set B is found using Graham scan [17] over the points of $\hat{P}$ projected onto the plane.

In this work, we assume the pose of the depth camera sensor has already been registered with respect to a global reference frame. This could be done (for example) using one of the variants of iterative closest point (ICP) based approaches [18]. After pose registration, in order to construct a meaningful geometric representation of a scene, it is necessary to merge the observed polygons from multiple scenes to form a single, "global scene". This is done in two steps:

1) Correspondence Matching: Polygons in the latest frame are matched to polygons in the global scene.
2) Polygon Merging: Matched polygons are merged to update the plane parameters and the convex hull of the polygons in the scene.

B. Correspondence Matching

Given a set $M = \{p_i^M\}$ of polygons that represent the map of polygons constructed so far and the set $O = \{p_j^O\}$ of observed polygons in the latest depth image, the correspondence matching problem is to find out which polygons p_j^O in the current depth image overlap with which polygons p_i^M in the map. In our approach, we solve this problem using ray casting:

1) A unique color is assigned to each polygon p_i^M in the map M.
2) A virtual camera with the same optical parameters (pixel size, field of view) as the actual depth camera is set up at the location found by pose registration.
3) The color-coded polygons are then rendered to an image as viewed by the virtual camera.
4) The pixel location of each plane filtered point in the depth image is then used to look up the color of the corresponding pixel in the rendered image, thus indicating the map polygon corresponding to that point.

The image rendering step, when performed using OpenGL and accelerated by an nVidia GeForce GTX 560 GPU, runs at an average of 3800 frames per second. After estimating observation to map polygon correspondences, these correspondences are further checked to ensure that matched polygons have an offset distance of less than η, and that the angle difference between their normals is at most θ. η and θ are configurable parameters, and in our experiments are set to 0.01m and 1.0° respectively. Observed polygons p_j^O that do not correspond to any map polygon are assumed to be new polygons, and added to the map M without merging with any existing polygon.

C. Polygon Merging

When merging the latest depth image observation with the map built so far, the polygons extracted from the latest depth image need to be merged with the corresponding polygons in the map built so far. The problem of polygon merging is

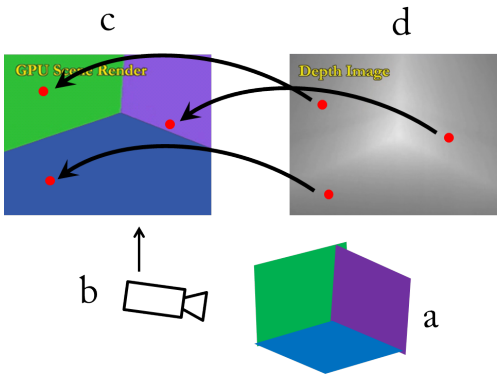


Fig. 2: Correspondence matching: Polygons in the map are uniquely color-coded (a), and from the perspective of a virtual camera (b) rendered in an image (c). Correspondences of pixels in the observed depth image (d) are then determined by inspecting the color of the same pixel in the rendered image.

thus: given two convex polygons c^1, c^2, we wish to construct a merged polygon c^m using the 3D points from both planes c^1 and c^2.

A naïve approach to merging polygons $c^1 = \{\hat{P}^1, n^1, \bar{p}^1, r^1, b_1^1, b_2^1, B^1\}$ and $c^2 = \{\hat{P}^2, n^2, \bar{p}^2, r^2, b_2^2, b_2^2, B^2\}$ to form merged polygon $c^m = \{\hat{P}^m, n^m, \bar{p}^m, r^m, b_m^m, b_2^m, B^m\}$ is to combine the 3D points as $\hat{P}^m = \hat{P}^1 \bigcup \hat{P}^2$ and then compute all other polygon parameters from $\hat{P}^m$. In practice, this is infeasible since it requires a complete history of all 3D points ever observed to be maintained and the amount of available memory is finite.

It is, however, possible to perform polygon merging without maintaining a complete history of all observed 3D points by in a manner analogous to the parallel computation of covariance matrices [19]. Scatter matrices from two different polygons cannot be directly combined due to the coupling in S between the points p_i and the centroid. However, S^1 may be decoupled as $S^1 = S_1^1 - n^1 S_2^1$ where S_1^1 depends on the 3D points in $\hat{P}^1$, and S_2^1 on the centroid $\bar{p}^1$. S_1^1 and S_2^1 are then given by $S_1^1 = \sum p_i^1 p_i^{1T}$ and $S_2^1 = \bar{p}^1 \bar{p}^{1T}$. Given the centroids $\bar{p}^1$ and $\bar{p}^2$ of the two polygons, the centroid $\bar{p}^m$ of the merged polygon is computed as their weighted mean. Thus, the matrix S_2^m of the merged polygon can be computed from $\bar{p}^m$ and the scatter matrix of the combined polygon is given by $S^m = S_1^1 + S_1^2 + n^m S_2^m$.

Therefore, the matrix S^1 along with centroid $\bar{p}$ and the number of points n is sufficient for merging polygons based on the complete history of observed 3D points, and the individual 3D points no longer need to be maintained for each polygon.

One implementation detail remains: to ensure numerical stability over time, the algorithm performs the eigenvector decomposition on the normalized matrix $\frac{1}{n}S$, and maintains normalized matrices $\frac{1}{n}S_1$ instead of S_1.

The convex boundary point set B^m of the merged polygon is computed by running Graham scan on the union of the points from the boundary point sets B^1 and B^2 of the constituent polygons.

The complete set of steps of our polygon merging algorithm are thus:

1) Compute centroid $\bar{p}^m$
2) Compute scatter matrix S^m
3) Compute normal r^m and basis vectors b_1^m, b_2^m by eigenvector decomposition of S^m
4) Compute boundary point set B^m using Graham scan on plane projected points from $B^1 \bigcup B^2$

The merged polygons in the map are thus updated using all observed points so far that correspond to them.

IV. Experimental Results

We performed two sets of experiments to demonstrate the computational efficiency and effectiveness of FSPF and the polygon merging algorithms. First, we collected data for a controlled scene with and without clutter, with ground truth. This ground truth data evaluates the accuracy of the merged polygons, both with respect to the observed depth image, as well as with respect to the measured ground truth. We then tested the FSPF and polygon merging algorithms on a set of real indoor scenes.

A. Ground Truth Comparison

We constructed a scene with planar features (Fig. 4a) including planes at various angles and different levels of occlusion. We then recorded depth image data from the scene and built a map of the scene using our algorithm. The errors in raw observation were evaluated by comparing the reconstructed 3D points with the ground truth values. We refer to this error comparison as the "point to ground truth" (PG) error. Fig. 4c shows the point to ground truth errors in observation for different parts of the image. Fig. 4b shows the map of polygons reconstructed from the scene. Except for the top surfaces of the two boxes on the right, all the dominant planes in the image are extracted by the algorithm. Fig. 4d shows the errors between the 3D points reconstructed from the depth image and the built map. We refer to this error comparison as the "point to map" (PM) error.

To evaluate the accuracy of the polygons in the built map, points on the polygons are uniformly sampled at intervals of $0.5cm$, and these 3D points are compared to the ground truth. We refer to this error comparison as the "map to ground

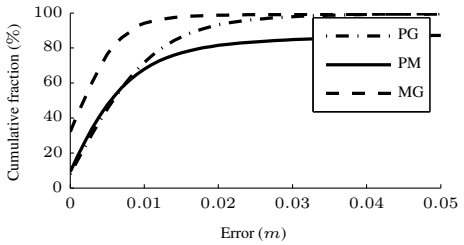


Fig. 3: Cumulative histogram of errors in mapping the clutter-free scene: (PG) depth image *vs.* ground truth, (PM) depth image *vs.* built map and (MG) built map *vs.* ground truth.

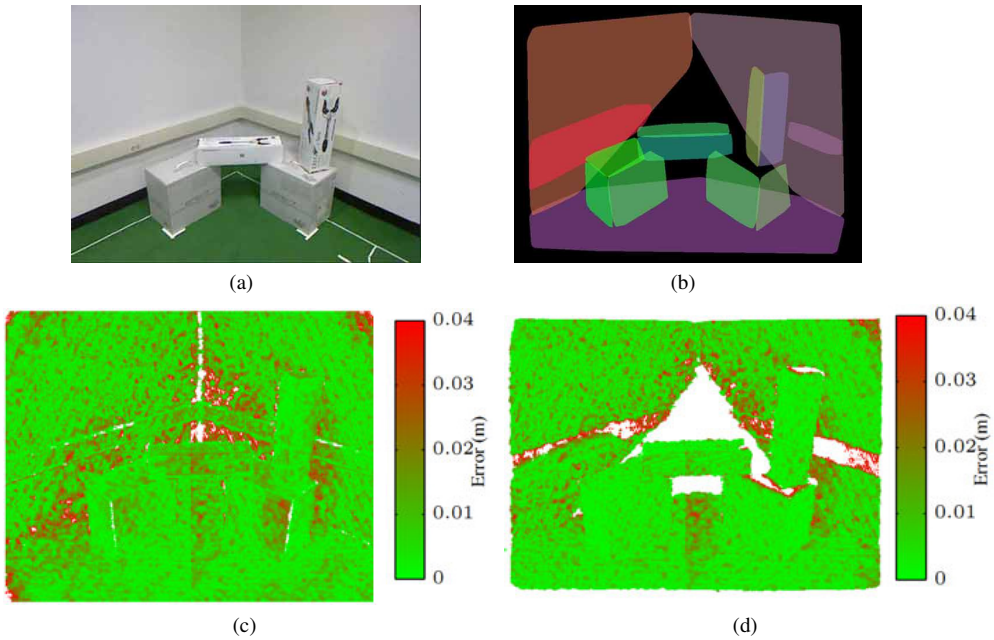


Fig. 4: Ground truth in a clutter-free scene: (a) RGB Image of the scene , (b) The map built by the polygon extraction and merging algorithm, (c) Errors between points in depth image compared to ground truth, and (d) compared to built map.

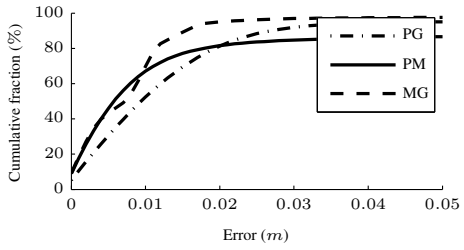


Fig. 5: Cumulative histogram of errors in mapping the scene with clutter: (PG) depth image *vs.* ground truth, (PM) depth image *vs.* built map, and (MG) built map *vs.* ground truth

truth" (MG) error. Fig. 3 shows a cumulative histogram of the three error comparisons.

We next evaluate the effect on the error in polygon map building due to the presence of clutter in the scene. Fig. 6a shows a controlled scene with added objects of clutter. Fig. 6c shows that the point to ground truth error is noticeably large in the regions of the scene corresponding to the objects of clutter. Fig. 6d shows that the built map(Fig. 6b) has lower errors than the point to ground truth case, but parts of the scene are unmapped. In spite of the local occlusions due to clutter on the boxes, the surfaces of the boxes are still reconstructed by the FSPF and polygon merging algorithms, and as Fig. 5 shows, the polygons that are constructed in the scene are still observed to have a maximum error of $2cm$.

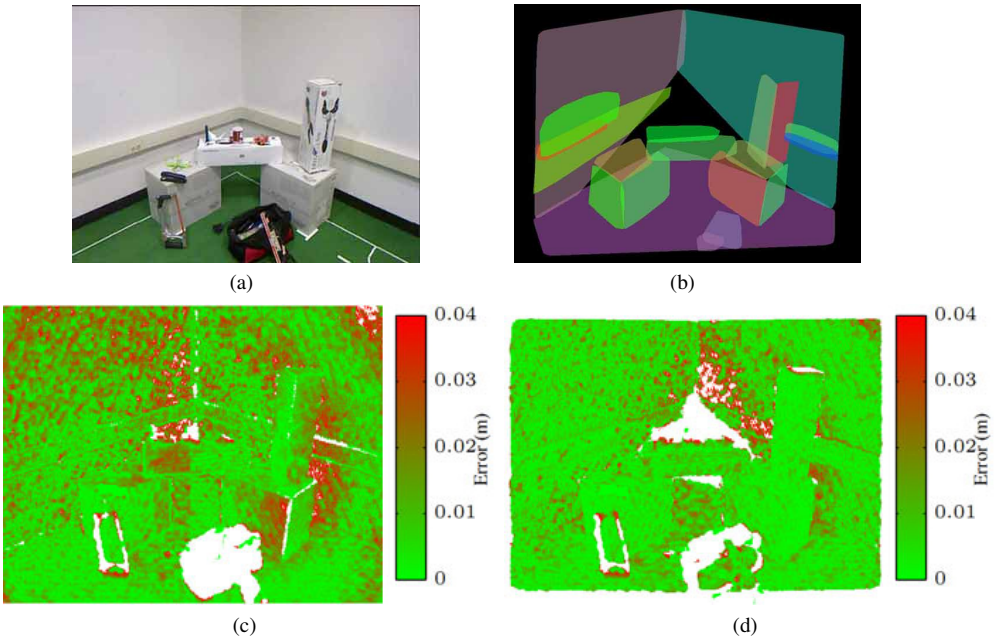
B. Real World Scene Tests

The FSPF and polygon merge algorithms were tested on data collected from 5 scenes that we selected as representative of our environment. At each scene, the Kinect sensor was kept static, so no pose update was necessary. For each scene, the polygons from 15 consecutive frames were merged to generate a set of polygons that represent the entire scene. Fig. 7 shows the scenes and the processed output from each scene, while Table I summarizes the output and processing times for the algorithms for each scene. The sampling efficiency for each scene is computed as the ratio of the mean number of plane filtered points per frame to the mean sampled locations per frame. The FSPF run time is per frame, and the polygon merge processing time is for all the polygons per frame.

Scene 1 has three boxes in the corner of a room. The merged scene polygons correctly include all the planes in the scene except two (the tops of the boxes on the left and center). Scene 2 is set in a corridor, and lacks clutter. The merged scene polygons correctly include all the polygons in the scene near the Kinect. Scene 3 is set in an open area next to a staircase. The merged scene polygons include the ground polygon, the side of the staircase, and the overhanging staircase. Scenes 4 and 5 show work areas with chairs and tables. Despite the clutter, all table surfaces are correctly identified among the merged scene polygons. Scene 5 had a toy chair on top of the table, and the merged scene includes its corresponding polygons as well.

V. Conclusion

In this paper, we introduced algorithms to generate convex polygons from single depth images, and to merge the polygons over time based. We experimentally showed the computational efficiency of FSPF and polygon merge algorithms. and evaluated their accuracy based on ground truth.



(a) (b) (c) (d)

Fig. 6: Ground truth in a scene with clutter: (a) RGB Image of the scene , (b) The map built by the polygon extraction and merging algorithm, (c) Errors between points in depth image compared to ground truth, and (d) compared to built map.

Scene Scene	Merged Polygons	Plane Filtered Points Per Frame	Sampled Points Per Frame	Sampling Efficiency (%)	Polygons Per Frame	FSPF Run Time (ms)	Polygon Merge Time (ms)
1	14	2002	5434	36.84	71	1.86	0.37
2	5	2006	8620	23.28	81	1.66	0.71
3	3	2001	11009	18.18	82	2.06	0.38
4	14	2004	8260	24.27	70	1.39	1.12
5	7	2001	6861	19.17	67	1.66	0.95

TABLE I: Results from test scenes

REFERENCES

[1] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3d modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics*, 2010.

[2] R.A. Newcombe and A.J. Davison. Live dense reconstruction with a single moving camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1498–1505. IEEE, 2010.

[3] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998.

[4] J. Biswas and M. Veloso. Depth Camera Based Indoor Mobile Robot Localization and Navigation. In *Robotics and Automation, 2012. ICRA'12. IEEE International Conference on*, pages 1697–1702. IEEE.

[5] N.J. Mitra and A. Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 322–328. ACM, 2003.

[6] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for Point-Cloud Shape Detection. In *Computer Graphics Forum*, volume 26, pages 214–226. Wiley Online Library, 2007.

[7] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak. Fast plane detection and polygonalization in noisy 3D range images. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3378–3383. IEEE, 2008.

[8] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[9] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *ICAR'05*, pages 242–249. IEEE.

[10] M. Gary, N. Fairfield, W.C. Stone, D. Wettergreen, GA Kantor, and J.M. Sharp Jr. 3D mapping and characterization of sistema Zacatón from DEPTHX (DEep Phreatic THermal eXplorer). In *Proceedings of KARST08: 11th Sinkhole Conference ASCE*. Citeseer, 2008.

[11] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 321–328. IEEE, 2000.

[12] J. Weingarten and R. Siegwart. 3D SLAM using planar segments. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3062–3067. IEEE, 2006.

[13] P. Kohlhepp, P. Pozzo, M. Walther, and R. Dillmann. Sequential 3D-SLAM for mobile action planning. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 722–729. IEEE, 2004.

[14] G. Vosselman, B.G.H. Gorte, G. Sithole, and T. Rabbani. Recognising structure in laser scanner point clouds. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 46(8):33–38, 2004.

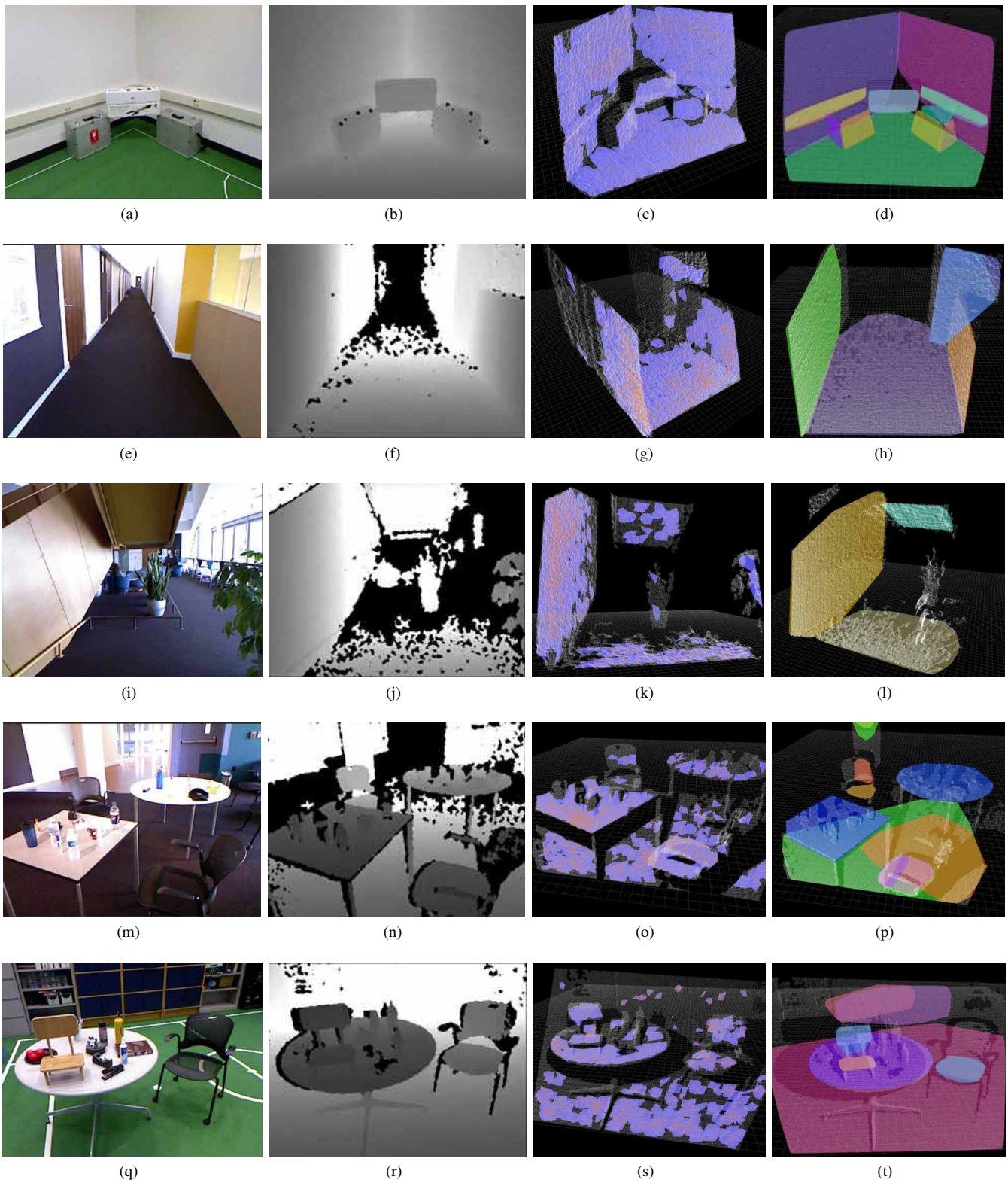
[15] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga. Online three-dimensional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation. *Journal of Field Robotics*, 27(1):52–84, 2010.

[16] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 127–136. IEEE, 2011.

[17] R. L. Graham. An efficient algorith for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132 – 133.

[18] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001.

[19] TF Chan, GH Golub, and RJ LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. *COMPSTAT 1982. Pt. 1. Proceedings*, 5:30, 1982.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

(q) (r) (s) (t)

Fig. 7: Test scenes 1-5 (top to bottom). First column shows the RGB image of the scene, second column the depth image, third column the plane filtered points (orange points) and polygons (lilac) for a single depth image frame, fourth column the polygons generated by merging 15 consecutive processed depth image frames. Columns 2 and 3 include overlays of the raw 3D point cloud (translucent white) for reference.